Removing Local Extrema from Imprecise Terrains

Chris Gray* Frank Kammer† Maarten Löffler‡ Rodrigo I. Silveira§

Abstract

In this paper we consider imprecise terrains, that is, triangulated terrains with a vertical error interval in the vertices. In particular, we study the problem of removing as many local extrema (minima and maxima) as possible from the terrain. We show that removing only minima or only maxima can be done optimally in $O(n \log n)$ time, for a terrain with n vertices. However, removing both the minima and maxima simultaneously is NP-hard, and is even hard to approximate within a factor of $O(\log \log n)$. Moreover, we describe an interesting connection between this problem and a graph problem that is a special case of 2-Disjoint Connected Subgraphs, a problem that has lately received considerable attention from the graph-algorithms community.

1 Introduction

1.1 Imprecise terrains

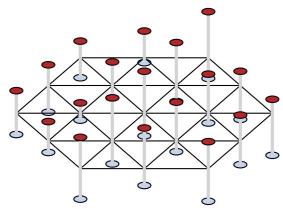
A triangulated (or polyhedral) terrain is a planar triangulation with a height associated with each vertex. This results in a bivariate and continuous function, defining a surface that is often called a 2.5-dimensional (or 2.5D) terrain.

Even though in computational geometry it is usually assumed that the input data is exact, in practice, terrain data is most of the time imprecise. The sources of imprecision are many, starting with the methods used to acquire the data, which are ultimately based on error-prone measuring devices. Often such methods produce heights with a known error bound or return a height interval rather than a fixed height value.

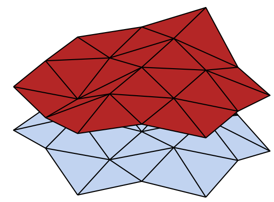
In order to model the imprecision in the terrain, we follow the model used in [4, 5, 8], where the height of each terrain vertex is not precisely known, but only an interval of possible heights is available. This results in considerable freedom in the terrain, since the "real" terrain is unknown and any choice of a height for each vertex—as long as it is within its height interval—leads to a valid *realization* of the imprecise terrain. The large number of different realizations of an imprecise terrain leads naturally to the problem of finding one that is best according to some criterion or that removes a certain type of unwanted feature (that is, an artifact).

We note that, even though terrain data may contain error also in the x, y-coordinates, under this model we consider imprecision only in the z-coordinate. This simplifying assumption is justified by the fact that error in the x, y-coordinates will most likely produce elevation error. Moreover, often the data provided by commercial terrain data suppliers only reports the elevation error [3].

*Department of Computer Science, TU Braunschweig, Germany, gray@ibr.cs.tu-bs.de
†Institut für Informatik, Universität Augsburg, Germany, kammer@informatik.uni-augsburg.de
‡Computer Science Department, University of California, Irvine, USA, mloffler@uci.edu
§Dept. de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain, rodrigo.silveira@upc.edu



(a)



(b)

Fig. 1: (a) An example of an imprecise terrain. (b) The same terrain, shown by drawing the floor and the ceiling.

In the remainder of this paper, an *imprecise terrain* is given by a set of n vertical intervals in $\mathbb{R}^3$, with a triangulation of the projection. See Figure 1(a). A *realization* of an imprecise terrain is a triangulated terrain that has the same triangulation in the projection, and exactly one vertex on each interval. An alternative way to view an imprecise terrain is by connecting the tops of all intervals into a terrain, which we call the *ceiling*, and the bottoms into a second terrain, which we call the *floor*. Then, a realization is a terrain that lives in the space left open between the floor and the ceiling. Figure 1(b) shows this in the example.

1.2 Removing local extrema

In this paper, we attempt to solve the *minimizing-minima*, the *minimizing-maxima*, and the *minimizing-extrema problem on imprecise terrains*, i.e., we attempt to find a realization of an imprecise terrain (by placing the imprecise points within their intervals) that minimizes the number of *local minima*, *local maxima*, and *local extrema*, respectively. A local minimum (or pit) is usually defined as a point that is surrounded by higher points, or that has no lower neighboring point. A local maximum (or peak) is defined analogously, as a point surrounded by lower points or without higher neighbors.

When terrains are used for land erosion, landscape evolution, or hydrological studies, it is generally accepted that the majority of local extrema in the terrain model are spurious, caused by errors in the data or model production. A terrain model with many pits or peaks does not represent the terrain faithfully, and moreover, in the case of pits, it can create problems because water accumulates at them, affecting water flow routing simulations. For this reason the removal of local minima from terrain models is a standard preprocessing requirement for many uses of terrain models [15, 18].

In this paper we aim at finding a realization of an imprecise terrain which minimizes the number of local minima or local extrema (that is, number of minima + number of maxima).

It is important to note that a group of k connected vertices at the same height without any lower neighbor is considered to be only *one* local minimum. This is reasonable from the point of view of the application, and follows the definitions used in [14]. In Section 5 we discuss what the implications of this modelling choice are for our results.

A lot of research has been devoted to the problem of removing local minima from (precise) terrains, although most of the literature assumes a raster (grid) terrain (e.g. [11, 12, 18]). Only a few algorithms have been proposed for triangulated terrains, mainly in the context of optimal higher order Delaunay triangulations [2, 6]. In particular, Gudmundsson *et al.* [6] show that the optimal number of both local minima and local maxima can be removed from first-order Delaunay triangulations in $O(n \log n)$ time. More related to this paper, Silveira and Van Oostrum [14] study moving vertices vertically in order to remove all local minima with a minimum cost, but do not assume bounded intervals.

1.3 Results

We first study the problem of finding a realization of an imprecise terrain that minimizes the number of local minima (or local maxima). We present a relatively simple algorithm that removes local minima or local maxima optimally in $O(n \log n)$ time.

Then we turn our attention to removing both minima and maxima. In short, we show that the problem of minimizing the number of local minima plus local maxima (that is, minimizing the number of local extrema) is NP-hard to approximate. For this we present two different proofs. The first one shows that minimizing-maxima cannot be approximated within a factor $3/2$ by exploiting the connection between the minimizing-extrema problem and a problem on graphs related to the 2-Disjoint Connected Subgraphs problem. The latter problem has received quite some attention recently, and we consider the connection between both problems interesting on its own. Some related results regarding 2-Disjoint Connected Subgraphs that are needed for the proof are given in Section 3, whereas NP-hardness proof itself is presented in Section 4.2. Furthermore, in Section 4.3 we present a second proof, based on Set Cover, that shows that the minimizing-extrema problem is NP-hard to approximate in an imprecise terrain with n vertices within a factor of $O(\log \log n)$.

Finally, in Section 5 we discuss how certain types of input degeneracy influence the results presented in the previous sections.

2 Removing local minima

In this section we want to find the realization that has the smallest number of local minima. We propose an efficient algorithm based on the idea of selectively *flooding* parts of the terrain. The algorithm begins with all vertices as low as possible, and simulates flooding parts of the terrain.

Algorithm Conceptually, we raise all local minima as much as possible, that is, we raise each minimum and its neighbors as we meet them, merging minima as we go. The process stops when one of the vertices in a local minimum cannot be raised any further.

We sweep a plane vertically, starting at the lowest interval end and moving upwards in the z direction. As the plane moves up, it *pulls* some of the vertices with it, whose height is changing together with the plane. At any moment during the sweep, each vertex is in one of three states:

(i) Moving, if it is currently part of a local minimum, and is moving up together with the sweep plane.

(ii) Fixed at a height lower than the current one.

(iii) Unprocessed, if it has not been reached by the sweep plane yet.

As the sweep plane moves vertically up, we distinguish two types of events:

(i) The plane reaches the beginning (lowest end) of the interval of a vertex,

(ii) The plane reaches the end (highest end) of the interval of a vertex.

Let v denote the vertex whose interval just began or ended, and let h be the current height of the plane. Note that all fixed vertices are fixed at a height lower than h.[1]

[1] For simplicity we are assuming in this description that all interval heights are different. The removal of this assumption does not pose any problem for the algorithm.

An event of type (i) can create a number of situations.

If v has a neighbor that is already fixed, then v will never be a local minimum, thus v is fixed at its lowest possible height, h. Moreover, if some other neighbor of v is currently part of a local minimum (i.e. is moving), then all the vertices part of that local minimum become fixed at h, and automatically stop being a minimum. This occurs for each neighbor of v that is currently part of a local minimum.

If all neighbors of v are currently unprocessed, then v will become a new local minimum, and will start to move up together with the plane.

Finally, if no neighbor is fixed but some neighbor is moving, thus is part of a local minimum, then v will join that existing local minimum and also start to move up together with the plane (note that if there is more than one local minimum that becomes connected to v, at this step they all merge into one).

Events of type (ii), when an interval ends, are easier to handle. If v is fixed, nothing occurs. If v was moving, then it becomes fixed at h, and the same occurs to all the vertices of the local minimum that contains v. Thus the whole local minimum becomes fixed, and will be present in the final solution.

Correctness The correctness of the algorithm can be proved by induction on the steps (i.e. events) of the sweep (associated with exactly $2n$ height values). Let h_i denote the height of the plane at the ith event. Clearly, for $h = h_1$ the terrain processed has only one local minimum, comprised of the lowest vertex, which is optimal. Now assume that for $h = h_i$ the solution is optimal. That is, the number of local minima in the imprecise terrain resulting from cropping the original terrain by the plane at height h_i (that is, $T \cap (z \leq h_i)$) is minimum.

We analyze the type of event that can take place for $h = h_{i+1}$. Let v be the vertex whose interval is ending or beginning at $h = h_{i+1}$.

If the interval of v is ending, then v is part of a local minimum that will be fixed. Since this local minimum already existed in the previous step, and that solution was optimal by the inductive hypothesis, the current solution is also optimal.

If the interval of v is just starting, it is only necessary to argue about the optimality of the connected component (in the cropped terrain) that contains v. The other connected components are optimal due to the inductive hypothesis, because they have not changed by this event.

Consider first the case in which v is connected to at most one fixed (lower) local minimum. Then the connected component that contains v in the current cropped terrain will end up consisting of a single local minimum. Since every connected component has at least one local minimum, this is optimal for the component where containing v.

In case that v is connected to more than one lower local minimum, we note that none of them can be removed by connecting them to v, because the lowest possible position for v is at h_{i+1}, which is higher than all its neighbors (recall that, by construction, fixed local minima are at their highest possible height). Therefore in this case the number of local minima for the connected component that contains v stays the same, leading again to an optimal solution for that component. Therefore the current cropped terrain has the minimum possible number of local minima, and the correctness of the algorithm follows.

Running time Sorting the interval ends for the sweep requires $O(n \log n)$ time. The rest of the steps can be implemented in linear time as follows. We maintain a hierarchical representation of each moving local minimum (each minimum is represented by a list of vertices, or a list of merged local minima). At the same time, for each vertex we store a pointer to the local minimum containing it, if any. When the interval of a vertex begins or ends, we need to traverse its neighbors, to know which of the cases described before applies. That takes time proportional to the degree of the vertex. Since every vertex is processed twice (at the beginning and end of its interval), we spend linear time in total for such operations. When a vertex becomes fixed, then all local minima connected to it become fixed as well. In that case we have to fix each vertex contained in each minimum, which takes time proportional to the number of vertices to be fixed. Since every vertex is fixed at most once, this takes linear time for the whole sweep. Hence the sweep itself (excluding sorting) runs in linear time, and the total running time is $O(n \log n)$.

Theorem 1: The number of local minima (or local maxima) in an imprecise terrain with n vertices can be minimized in $O(n \log n)$ time.

It is interesting to note that when a group of k connected vertices at the same height without any lower neighbors is regarded as k different local minima, the problem can be proved NP-hard. More details on this are given in Section 5.

3 Intermezzo: Splitting Graphs

In order to prove our remaining results for imprecise terrains, we first show some results in a graph problem that turns out to be related to ours. As we show later, there is a strong connection between the problem of removing local extrema from imprecise terrains and a graph problem, which we call Planar 2-Disjoint Maximally Connected Subgraphs (or P2-MaxCon for short).

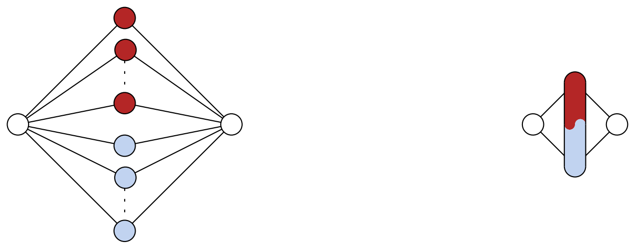
This problem is a special case of 2-Disjoint Connected Subgraphs (or 2-Con for short). In this problem, one is given a graph $G = (V, E)$ and two subsets $R \subset V$ and $B \subset V, R \cap B = \emptyset$ of vertices that are colored red and blue. The objective is to find two subsets $R' \supset R$ and $B' \supset B, R' \cap B' = \emptyset$ such that both (R', E) and (B', E) are connected graphs, that is, to color some of the remaining vertices red or blue to make both the red and the blue subgraph connected. Recently, Van 't Hof *et al.* [16], showed that 2-Con is already NP-hard when there are only two red vertices. Paulusma and Van Rooij [13] try to tackle the problem by designing more efficient exact algorithms. Kammer and Tholey [7] also study a related problem. However, there are, to our knowledge, no results on this problem for planar graphs. We define Planar 2-Disjoint Connected Subgraphs (or P2-Con for short) as the same problem as 2-Con, except that G is known to be planar. We also define Planar 2-Disjoint Maximally Connected Subgraphs as the optimization variant, where the goal is to optimise the number of connected components (red and blue together) in the output graph, rather than to require that both graphs are completely connected.

We now proceed to show that P2-MaxCon is NP-hard.

3.1 P2-MaxCon is NP-hard

We prove that P2-MaxCon is NP-hard by a reduction from planar 3-SAT [10]. In this problem, the normal 3-SAT problem is restricted so that the bipartite graph connecting variables and clauses is planar. We call this graph $G_S = ((V \cup C), E)$, where an edge $e = (v, c) \in E$ if variable v is in clause c. As is usual in such reductions, we first embed G_S in the plane so that none of the edges in E cross. We then replace the vertices and edges in the embedding with "gadgets".

The variable gadget is simply a white vertex. We show below that coloring the vertex red is equivalent to setting the corresponding variable to `true` and coloring the vertex blue is equivalent to setting the corresponding variable to `false`.



(a)

(b)

Fig. 2: (a) An inverter, consisting of k red and blue vertices. (b) Symbolic representation of an inverter.

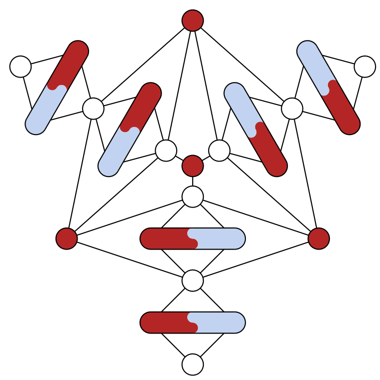


Fig. 3: In a clause, we connect three inverter gadgets using four extra red vertices.

Another gadget that we use is the inverter gadget, shown in Figure 2(a). This gadget consists of two white vertices, k red vertices, and k blue vertices. Each colored vertex is connected to both white vertices. This gadget ensures that one of the white vertices must be colored red and the other one blue, because otherwise there will be k components in the output. To ensure that this is unacceptable for any optimal solution, we make k at least as large as the number of gadgets that we use in our construction.

A clause gadget is a collection of three inverter gadgets, as well as four extra red vertices. These are all connected as shown in Figure 3. The red vertices form one large component as long as at least one of the white vertices adjacent to the central red vertex is colored red.

Finally, we create edge gadgets to connect variable gadgets to clause gadgets. An edge gadget is simply a chain of inverter gadgets. See Figure 4. If a variable v is negated in clause c, then we replace the edge (v, c) with a chain of an odd number of inverter gadgets, otherwise, we use an even-length chain. Since the number of inverter gadgets between a variable gadget and one of the clause gadgets that it is connected to determines the color of the final white vertex in the chain, we can see that coloring a variable gadget red corresponds to the final white vertex in a chain to a clause in which that vertex is not negated being colored red. This implies that coloring a variable gadget red is equivalent to setting its value to `true`, and that coloring a variable gadget blue is equivalent to setting its value to `false`.

The total number of connected components is equal to the number of white vertices in the construction, minus 2 per clause since the red components are connected, plus the number of unsatisfied clauses. Hence, minimizing the number of connected components involves determining whether the 3-SAT clause can be satisfied, which proves the following.

Theorem 2: P2-MaxCon is NP-hard.

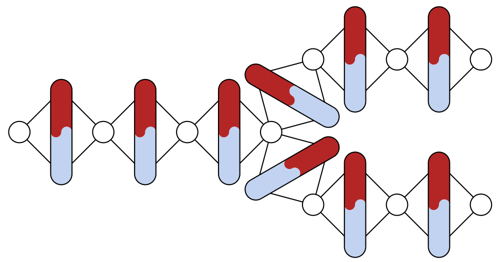


Fig. 4: We can chain inverter gadgets. The white vertices must always be colored alternately red and blue.

3.2 P2-Con is NP-hard

We now show how the construction above can be extended to show that also the more specialized problem P2-Con is NP-hard. The main difference is that we must ensure that at the end of the construction, all blue components become connected into one large blue component, and all red components become connected into one large red component.

First of all, we need some property of triangulated graphs. Let $G = (V, E)$ be a graph embedded in the plane such that all faces of G, except the outer face, are triangles. Let R be a subset of the vertices of G (say, the red vertices), such that no vertex of R is on the outer face of G. We say that the subgraph of G induced by R contains a *proper loop* if it contains a cycle that has at least one vertex of $V \setminus R$ inside. This leads to the following well-known observation. See, for example, the book by West [17] for a proof.

Observation 1: If the subgraph of G induced by R has no proper loops, then the subgraph of G induced by $V \setminus R$ is connected.

Now, let G again be a planar triangulated graph, and suppose that all vertices of G are colored either red or blue, so $V = R \cup B$, and suppose further that every red or blue component has at least one vertex on the outer face. Figure 5(a) shows such a graph. Then obviously neither R nor B has a proper loop.

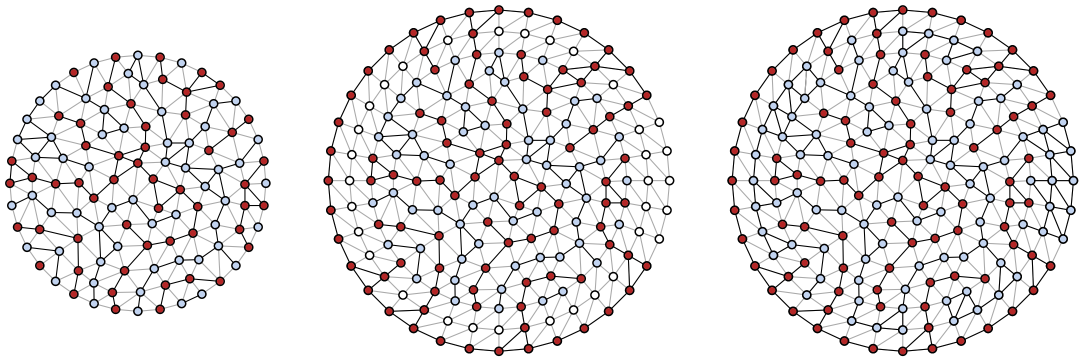
This means that whenever we have such a graph with a sufficient number of layers of white vertices around it, then we can color it such that we get only one large red component and only one large blue component.

Lemma 1: Let G be as before, and let G' be a larger graph that contains G and has 2 extra layers of white vertices around G, each at least as large as the outer face of G. Then we can color the white vertices of G' red or blue such that G' has only one red and one blue component.

Proof: We know that all components have at least one vertex on the outer face. For each red component, we select exactly one such vertex and color its counterpieces on the two extra layers also red. Then we color the vertices of the outer layer red to connect all the red components into one large component, as shown in Figure 5(b).

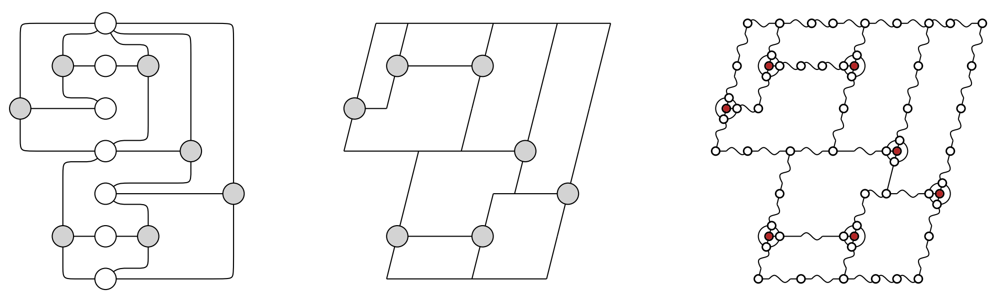
By doing this, we cannot create any proper loops because we only took one vertex from each red component, and because of the regular structure of the two outer layers. Therefore, the red component does not have proper loops, so by Observation 1 the complement is connected. Hence, we can color the complement blue to obtain a valid coloring. ⊠

We now show how the construction in the previous section can be extended to show that not only P2-MaxCon, but also P2-Con is NP-hard. To do this, we must make a construction such that, when the SAT formula is satisfiable, all red components can be connected into one large red



(a) (b) (c)

Fig. 5: (a) A triangulated graph, colored such that every component has a vertex on the outer face. (b) The same graph, augmented with two extra layers of white vertices. The red components are connected into a large components without proper loops. (d) All the remaining white vertices can be colored blue, making the blue component also connected.



(a) (b) (c)

Fig. 6: (a) A layout of a planar 3-SAT instance, where the variable nodes (white) are aligned on a single vertical line, and the clause nodes (gray) are on both sides of the line. (b) Polygonal subdivision into x-monotone polygons as a result of replacing the variable nodes and adjacent edges by polygonal trees. (c) The construction embedded onto the subdivision. The inverter gadgets are indicated by wiggling edges. Note that there is some freedom in the construction as to how many vertices and inverter gadgets are placed on the edges; only the parity matters.

component, and all the blue components can be connected into one large blue component. If the formula is not satisfiable, this should not be possible.

First, it is well known that a planar 3-SAT graph can be embedded in the plane having all variable nodes on one vertical line, and the clause nodes on both sides, as in Figure 6(a). From this, we observe that if we replace the variable nodes and incident edges by polygonal trees, we can make the embedding such that all faces of the graph are x-monotone polygons (except for the outer face), as shown in Figure 6(b) for the example graph. This implies a partial ordering on the faces of the graph, based on their above-below relation.[2]

We now replace the polygonal trees by chains of inverter gadgets, as in the previous section. We also replace the clause nodes by clause gadgets as before, except that we do not include the three extra red vertices. Instead, we connect the neighboring vertices of each clause. Figure 7 shows the simplified clause gadget. Figure 6(c) shows an example of the resulting embedding.

[2] In fact we don't really need them to be x-monotone polygons, any embedding with a directed dual graph that induces a partial ordering such that all faces at the top of the ordering are on the outside of the construction would be good enough for the argument.

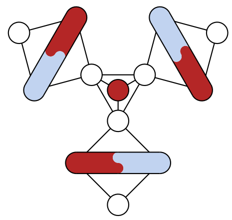


Fig. 7: A clause gadget, simplified.

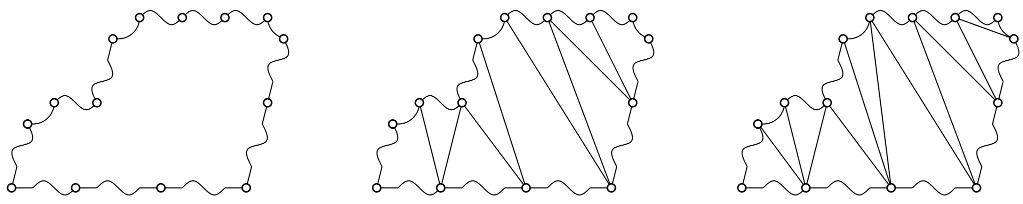


Fig. 8: (a) An x-monotone polygon with some white vertices and inverter gadgets on its boundary. (b) Each white vertex on the bottom has been connected to two white vertices at the top that are separated by an inverter gadget. (c) Some additional edges are inserted to make the graph formed by the white vertices triangulated.

The resulting construction has many white vertices on the boundaries of the x-monotone polygons, which end up being colored either red or blue. Furthermore, all components in a final coloring contain at least one of these white vertices, except for those in the unsatisfied clauses. So, what remains to be done is make sure that these white vertices can be connected into one large red and one large blue component, no matter how they are colored.

To do this, consider one x-monotone polygon and its white vertices, as in Figure 8(a). An x-monotone polygon has two x-monotone polygonal chains that both connect the leftmost point to the rightmost point. We assume that on both of these chains there are at least 2 white vertices. Furthermore, we assume that the top chain of any polygon has at least as many white vertices as the bottom chain. If any of these assumptions is not satisfied, we simply add more gadgets to the chains: adding two inverter gadgets into a chain does not change the properties of the construction, and the above/below relations of the polygons define a partial order on them so this process will end. We then add edges to the interior of the polygon, connecting every white vertex on the bottom chain to two adjacent white vertices on the top chain that are on both sides of an inverter gadget, as in Figure 8(b). This means that whatever the color of a vertex of the bottom chain is, it is always connected to at least one vertex of the same color on the top chain. By induction, this means that every white vertex in the whole construction will be connected to some vertex on the top of the construction that has the same color. Finally, we triangulate the polygon (or rather, the graph of the white vertices involved in the polygon) by adding arbitrary edges if necessary, see Figure 8(c).

To prove that the construction is indeed colorable with two components if the 3-SAT formula is satisfiable, first consider the coloring that makes all clause gadgets satisfied. Then remove the clause vertices (leaving only the three white vertices and the triangle of edges connecting them), and replace all inverter gadgets by edges. The resulting graph is triangulated, and by the above argument, each component is connected to at least one vertex on the outside. We can then apply Lemma 1 to show it can be colored. Conversely, if the 3-SAT formula is not satisfiable, it is not possible to color the construction with two colors such that the vertices with equal colors form two connected components because one of the clauses cannot be satisfied. This implies that at least one of the clause gadgets cannot be properly colored.

Therefore we have proved the following.

Theorem 3: P2-Con is NP-hard.

4 Removing all local extrema

We now move to the problem of removing both local minima and local maxima, that is, removing all local extrema at the same time. Although the algorithm in the previous section works for both removing minima and removing maxima, it is not possible to use both height assignments simultaneously. However, we will show in the next section that we can still use the algorithm twice to narrow down the problem, without changing the value of the solution.

Unfortunately, such an approach does not help to find an optimal realization minimizing the number of maxima. In Sections 4.2 and 4.3 we give two NP-hardness proof that show that minimizing-extrema is NP-hard to approximate. The first proof exploits the connection with the Planar 2-Disjoint Maximally Connected Subgraphs problem. The second proof, which shows inapproximability within a factor of $O(\log \log n)$, uses a reduction from Set Cover.

4.1 Narrowing down a solution by cleaning up the floor and ceiling

Recall that the floor F is the terrain formed by all lower endpoints of the imprecise vertices, and the ceiling C is the terrain formed by all upper endpoints, as shown in Figure 1(b). We are searching for a surface between the floor and the ceiling that optimizes the number of local extrema. We will run the algorithm in the precious section on (F, C) to remove the local minima, and call the result F', and run it again on (F, C) to remove local maxima and call the result C'.

Lemma 2: The imprecise terrain induced by (F', C') is still a valid imprecise terrain which has the same optimal solution for removing local extrema as the original terrain (F, C).

Proof: We need to show two things. To show that (F', C') is still a valid imprecise terrain, we need that the height of any vertex in F' is at least the height of that vertex in C'. This is true because a plateau rising the floor and one lowering the ceiling of the same vertex could never have crossed each other.

To show that (F', C') has the same optimal solution as (F, C), we need to show that there exists an optimal terrain T^* between F and C that in fact also lies between F' and C'. This is true because if a terrain would have a local minimum below F', we could freely lift it together with its neighbors until it coincides with F'. By the construction of F', we will never hit the ceiling during this process, so we will never increase the number of minima or maxima. The converse is true for local maxima above C'. ⊠

Lemma 2 implies that we can run the algorithm of Section 2 as a preprocessing step, while still allowing a solution as good as in the original problem. Furthermore, this simplified problem has more structure than the original one. Every remaining local minimum of the floor touches the ceiling, and every remaining local maximum of the ceiling touches the floor. We can show the following:

Lemma 3: The total number of extrema in the optimal solution T^* is never greater than the number of local maxima of the floor F' + the number of local minima of the ceiling C'.

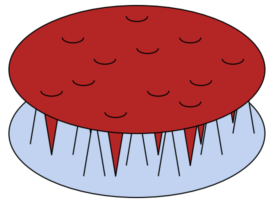


Fig. 9: An imprecise terrain that has many maxima on the floor and many minima on the ceiling.

Proof: Consider the solution $T = C'$. Any local maximum of T must touch a unique local maximum of F', because when lowering the maxima of C this was exactly the condition on which we stopped. Therefore, the number of local maxima of T is smaller than the number of local maxima of F'. Thus, the number of local extrema in T is at most the number of local maxima of F' + the number of local minima of C' (which is T itself). Clearly, the number of local extrema in the optimal solution T^* can only be even smaller. ⊠

If we denote by $\mathrm{lmin}(T)$ the number of local minima in a terrain T and by $\mathrm{lmax}(T)$ the number of local maxima in T, and we denote by T^* the optimal solution of our problem (F, C), then we can summarize these observations as follows:

$$\mathrm{lmin}(F') + \mathrm{lmax}(C') \leq \mathrm{lmin}(T^*) + \mathrm{lmax}(T^*) \leq \mathrm{lmin}(C') + \mathrm{lmax}(F') \tag{1}$$

This formula gives a bound on the values of a particular instance. In theory, the gap may still be arbitrarily large. For example, consider an instance where the floor is more or less flat except for a number of “stalagmites” that reach all the way to the ceiling, and the ceiling is more or less flat except for a number of “stalactites” that reach all the way to the floor. Figure 9 show such a situation. In this case, the number of maxima of the floor and minima of the ceiling is large, while the floor has only a single minimum and the ceiling has only a single maximum, and the preprocessing step will not make a difference. We see in the next section that this makes the problem very hard to solve.

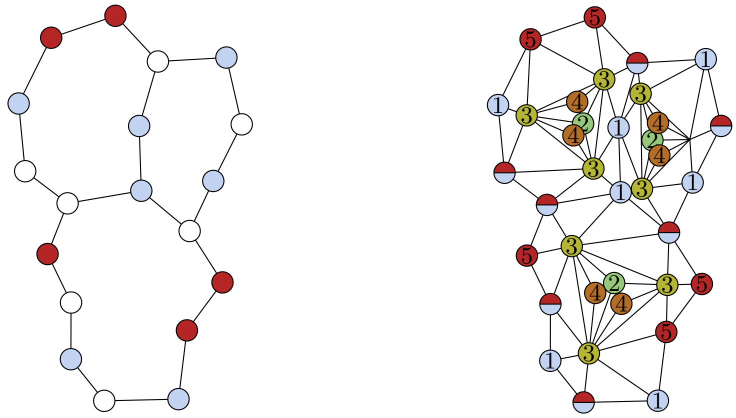
On the other hand, such terrains seem unlikely to appear in real applications. It may be likely that in practice, the gap in Equation 1 is quite small. Under which properties of terrains this is the case remains an interesting open question.

4.2 P2-MaxCon reduces to minimizing-extrema

We now show that the relation between the problem of removing local extrema from imprecise terrains and the graph problem P2-MaxCon implies that minimizing the number of local extrema in an imprecise terrain is NP-hard. The idea is to take an instance of P2-MaxCon, and construct from it an imprecise terrain similar to the one shown in Figure 9, by replacing the red vertices by stalactites and the blue vertices by stalactites, and the white vertices by open space.

In our reduction, we take the input to P2-MaxCon—a planar graph with red, blue and white vertices—and build an imprecise terrain from it. We will first embed the graph in the plane with straight edges. We then turn all red vertices into precise vertices at height 5, and all blue vertices into precise vertices at height 1. Finally, we turn the white vertices into imprecise vertices with interval $[1, 5]$.

The problem of minimizing extrema on this graph is equivalent to that of minimizing connected components after recoloring. This is due to the fact that the only way to remove local minima in this terrain is by connecting the minima to each other by putting the white vertices at height 1. Similarly, the only way to remove maxima is to put the white vertices at height 5.



(a) (b)

Fig. 10: (a) An instance of P2-MaxCon. (b) In the output, we fixed the red vertices at height 5, and the blue at height 1. The vertices with two colours are imprecise vertices with interval $[1, 5]$. The rest of the vertices are added to make sure that the graph is triangulated, and that the new vertices do not interfere with the number of local extrema.

However, to have a proper imprecise terrain, we must triangulate the graph. To do this, we add extra vertices and edges as shown in Figure 10(b). This adds a component with three extra minima and maxima per inner face of the graph. In this way, all previous (red, blue, white) vertices are connected to new vertices at height 3. These cannot help the red or blue vertices to stop being extrema, and at the same time cannot be extrema because they are connected to a lower and higher vertex inside the component (at heights 2 and 4). Given a solution that minimizes the number of extrema in the imprecise terrain that we have constructed, we can color the white vertices either red or blue. For any white vertex whose height is set to 1, we set the color to blue, otherwise we set the color to red.

Since P2-MaxCon is NP-hard by Theorem 2, we conclude that minimizing the number of extrema in an imprecise terrain is also NP-hard. In fact, by Theorem 3, P2-Con is also NP-hard. This implies that we cannot even approximate the number of local extrema better than in a factor $3/2$, because an optimal solution has at least 2 local extrema and a non-optimal solution must have at least 3.

Theorem 4: Minimizing the number of local extrema in an imprecise terrain is NP-hard to proximate within a factor $3/2$.

4.3 More hardness of approximation

In this section we show by a reduction from the Set Cover problem that we cannot approximate the number of local extrema on n-vertex graphs within any factor better than $O(\log \log n)$ unless $\mathrm{P} = \mathrm{NP}$. Given a tuple (U, C), where U is a finite set called *universe* and $\mathcal{C}$ is a collection of subsets of U, a *set cover* for (U, C) is a collection $C' \subseteq C$ such that the union of all sets in C' is equal to U. The *size* of C' is its cardinality. The Set Cover problem is to find a set cover of minimal size.

Let (U, C) be an instance of the Set Cover problem. We start by defining a graph G with colored vertices. We then construct a terrain by embedding G in the plane, and triangulating its faces with more than 3 incident vertices. Finally, we assign heights to the terrain vertices depending on their colors.

We begin by describing the top of the constructed graph, which consists in a *top gadget* depicted in Figure 11. For each item x of the universe U, we introduce $|U|+3$ red vertices with a blue vertex between each pair of red vertices. All blue vertices are connected to another blue vertex $v^{\min}$ at the very top of the construction. Each vertex $v \neq v^{\min}$ in the top gadget is the beginning of a path—that we call *downward path* (indicated in the figures by dashed arrows). Moreover, downward paths are marked as either *covered* or *uncovered*. At this stage, all downward paths starting with a red vertex are considered uncovered.

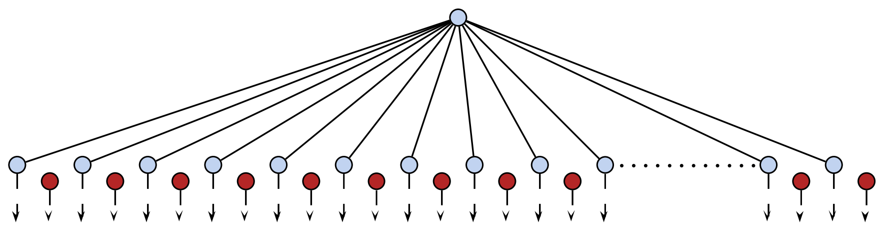


Fig. 11: Top gadget.

The construction continues by adding one *row-gadget* for each set $S \in C$ (see Figure 12 for a schematic representation). Each row-gadget extends the downward path downwards and consists of a row of vertices that we call the *decision row*. The leftmost and rightmost vertex of each decision row are, in fact, the same—the edges connected to these vertices meet in the space above the top of the currently-constructed graph. Every decision row consists of white vertices that must be assigned a color. To ensure that the colors assigned to the white vertices alternate between blue and non-blue, we place inverter gadgets as shown in Figure 13(a) between every pair of white vertices in a decision row. Additionally, each row-gadget contains one yellow vertex and several subgadgets that we describe later.

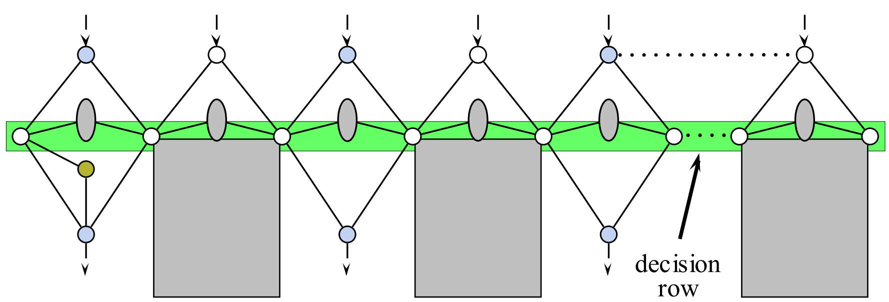


Fig. 12: Row-gadget for a set of C. Each gray box contains a subgadget.

As shown in Figure 12, the subgadgets always have a white vertex above them. Depending on the situation of that white vertex, we distinguish two different kinds of subgadgets.

If the white vertex above a subgadget is part of an uncovered downward-path or is part of a downward-path for an item $x \notin S$, we use the subgadget of Figure 13(b). A downward-path exits the subgadget in Figure 13(b) marked as covered if and only if it entered this subgadget marked covered.

Otherwise, we use the subgadget shown in Figure 13(c). Note that the downward path going through the white vertex above such subgadget either continues its way by the left or right white vertex in the subgadget. The other white vertex is the first vertex of a new downward path—this new downward path is *free*, that is, it does not correspond to an item in U. We mark the left downward path as covered, and the right downward path as uncovered.

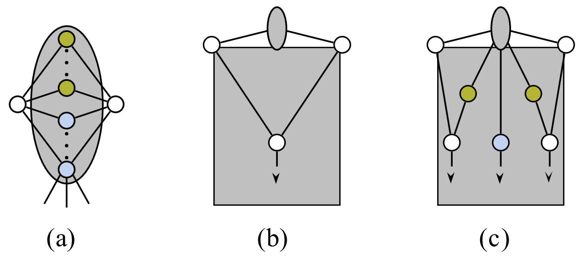


Fig. 13: (a) Inverter gadget consisting of $|U| + 3$ blue and yellow vertices. (b) and (c): Subgadget part of a gadget for a set S in C.

The construction ends with a *bottom gadget*, which is more or less symmetric to the top gadget, see Figure 14. The lowest vertex $v^{\max}$ is red. The color of the vertices with the two colors in Figure 14 is decided with the following rule. If such a vertex is the end of a path that is marked uncovered, then it is colored yellow. Otherwise, it is colored red.

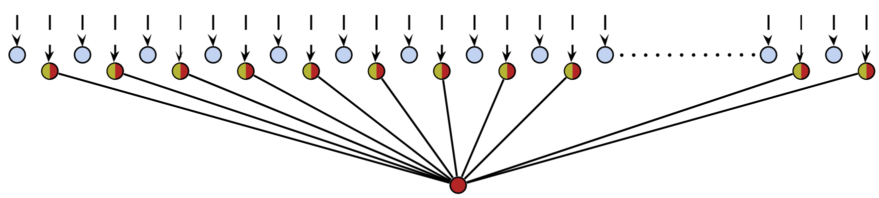


Fig. 14: Bottom gadget. A two-colored vertex is colored either red or yellow depending on whether it is the endpoint of a path marked covered or uncovered.

Let G be the graph obtained with straight-line embedding φ. The heights are assigned to vertices as follows. For simplicity, we use colors to refer to vertices with the same (imprecise) height. Vertices colored red have height 5, yellow vertices have height 3, blue vertices have height 1, and white vertices have a height in the range $[1, 5]$. To triangulate G, we add a vertex v_F of height 2 into each face F of φ and connect v_F to all vertices adjacent to F in φ. Let T be the set of vertices added during the triangulation.

To see that the size of our reduction is polynomial, we must show that the number n of vertices of G is polynomial in $|C|$ and $|U|$ since this graph is planar and thus $|T|$ is linear in n. Note that a row-gadget H for a set being crossed by z downward paths consists, for each downward path, of an inverter gadget and a constant number of further vertices. Thus, H has $O(z|U|)$ vertices. Moreover, by induction it is easy to observe that in the top gadget we have $O(|U|^2)$ downward paths and below the i-th row-gadget for a set, we have at most $O((i + 1)|U|^2)$ downward paths. This observation implies that we have at most $O(|C||U|^2)$ downward paths and all subgadgets for a set in C have in total $O(|C|^2|U|^2)$ vertices. Since the top and the bottom gadgets have fewer vertices, the graph obtained has $n = O(|C|^3|U|^2)$ vertices.

To minimize the number of local extrema, we color the vertices so that all blue vertices are one connected component and all red vertices are one connected component. Moreover, the yellow vertices are connected either to a vertex of the same kind or connected to both a blue and a red vertex.

Theorem 5: The minimizing-extrema problem in an imprecise terrain with n vertices cannot be approximated within a factor of $O(\log \log n)$ in polynomial time, unless $\mathrm{P} = \mathrm{NP}$.

Proof: We first show that each instance I_1 for the Set Cover problem of optimal cost $z - 2$ is reduced to an instance I_2 for the minimizing-extrema problem of optimal cost z such that each a solution for I_2 of cost y can be easily transformed into a solution for I_1 of cost $y-2$.

Let $C' \subseteq C$ be a set cover. A coloring of the graph can be found as follows. In each decision row, we color the vertices alternating in blue and non-blue (red or yellow). We choose for a vertex v between yellow and red depending on whether v is incident (from above) to a red vertex. If so, color v red, and otherwise yellow. Color the white vertices in the gadget of Fig. 13(c) by the same rule, in red or yellow. Additionally, we color the first vertex in a decision row red if and only if the decision row is part of a gadget for a set $S \in C'$.

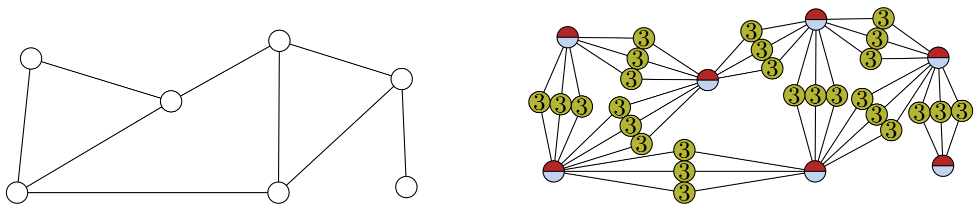
In this coloring, all vertices in T are connected to both a blue vertex and either a red or yellow vertex. Note that all yellow components are connected to a blue vertex and a red vertex. Moreover, all blue vertices form one connected component. By our choice of the height of two-colored vertices in the bottom gadget, the red vertices also induce a connected component since a path starting from a red vertex in the top-gadget contains only red vertices and ends in the bottom row in a red vertex. Apart from $v^{\min}$ and $v^{\max}$, the only local extrema that we have are created by the yellow vertices in the row-gadgets introduced for each set in C'. In other words, we have exactly $|C'|+2$ local extrema.

For the converse, let us first consider the case in which we have a solution with at least $|U|+2$ local extrema. We then claim that the optimal set cover contains all the sets of C. Now, let us assume that we have a solution of cost $z < |U|+2$. Then the vertices in each decision row are alternately colored in blue and non-blue, and there is a path for each item x of the universe U that is connected to $v^{\max}$. Our construction then implies that there is a set S in C with $x \in S$ such that the row-gadget for S has a local maximum at its yellow vertex. If we choose C' as the collection containing all sets of C whose row-gadget contains a local maximum at its yellow vertex, then C' is a cover of size at most $z-2$.

Alon, Moshkovitz, and Safra [1] showed that, for an appropriately chosen constant $c' > 0$, there is no polynomial-time approximation algorithm of approximation ratio $c' \ln |U|$ for the Set Cover problem with universe U unless $\mathrm{P} = \mathrm{NP}$. Since each set-cover instance with an optimal solution of cost 1 can be solved to optimality in $|U|^{O(1)}$ time, there can not exist a polynomial-time approximation of approximation ratio $c' \ln |U|$ for the Set Cover problem restricted to instances with optimal solutions of cost at least 2 unless $\mathrm{P} = \mathrm{NP}$.

Assume for contradiction that an approximation algorithm exists for the minimizing-extrema problem in an imprecise terrain with n vertices with an approximation ratio $(c'/8)\ln\ln n$. This means, each instance of the minimizing-extrema problem of optimal cost x' can be solved with cost at most $((c'/8)\ln\ln n)x'$. By our reduction from above, each set-cover instance $I_1 = (U, C)$ with optimal cost $x \geq 2$ can be first transformed into an instance I_2 of the minimizing-extrema problem with $n \leq |C|^3|U|^2 \leq (2^{|U|})^3|U|^2 \leq 18^{|U|}$ vertices and with optimal cost $x' = x + 2 \leq 2x$. By our assumption, we can solve I_2 such that the obtained solution has cost at most $((c'/8)\ln\ln n)x' \leq ((c'/4)\ln\ln n)x$, that is, we can solve I_1 with cost at most $((c'/4)\ln\ln n)x - 2 \leq ((c'/4)\ln\ln 18^{|U|})x \leq (c'\ln|U|)x$—for the latter inequality we use the fact that the cost of an optimal solution is at least 2, i.e., $|U| \geq 2$. This is a contradiction to the last paragraph. Thus, there can not exist an approximation algorithm for the minimizing-extrema problem in an imprecise terrain with n vertices with an approximation ratio $(c'/8)\ln\ln n$. ⊠

Note that Kumar, Arya, and Ramesh [9] showed that the Set Cover problem with universe U cannot be approximated within a factor of $o(\log |U|)$ in random polynomial time unless $\mathrm{NP} \subseteq \mathrm{ZTIME}(n^{O(\log\log n)})$ even if we restrict the collections C such that $|S_1 \cap S_2| \leq 1$ for all $S_1 \neq S_2$ in C. This restriction to C means that $|C| \leq |U|^2$. To see this, let $U = \{u_1, \ldots, u_{|U|}\}$, and consider a subcollection C_1 of C, whose subsets all contain u_1. Then it is easy to see that the number of sets in C_1 with a fixed $u' \in U \setminus \{u_1\}$ in them is at most 1. Thus, $|C_1| \leq |U|$. Applying the same argument to all subcollections $C_2, \ldots C_{|U|}$ that have $u_2, \ldots, u_n$, respectively, in them we conclude that $|C| \leq |U|^2$.



(a) (b)

Fig. 15: (a) A planar graph (b) The white vertices are imprecise vertices at height $[1, 5]$, the edges have been replaced by groups of k yellow vertices, which are fixed at height 3.

Using the reduction from above the graph obtained for such a restricted set cover instance (U, C) has $n' = |U|^{O(1)}$ vertices; thus, we can conclude the following:

Corollary 1: The minimizing-extrema problem in an imprecise terrain with n vertices cannot be approximated within a factor of $o(\log n)$ in random polynomial time, unless $\mathrm{NP} \subseteq \mathrm{ZTIME}(n^{O(\log \log n)})$.

5 Degeneracy

We have shown that removing local minima is easy and removing local extrema is hard. However, some of our results are dependent on degeneracy issues. In this section we will describe how these issues influence our results.

There are two separate issues. One is how we treat vertices of the same height in a realization of an imprecise terrain (that is, in a precise terrain). The other is whether we allow the tops and bottoms of the intervals in the imprecise terrain to have duplicate heights, that is, whether the input is assumed to be in general position.

5.1 Removing minima is sometimes hard

In Section 2, we have shown that all local minima can be removed from an imprecise terrain in $O(n \log n)$ time. However, this result is based on the viewpoint that when a group of vertices all have the same height, we count them as a single minimum. This is a common viewpoint in the literature. Nonetheless, we show here that if we count them as individual local minima, and the input is not in general position, the problem becomes hard.

The reduction is from maximum independent set on planar graphs. Given an instance of maximum independent set (a planar graph), we build an imprecise terrain as follows. Each vertex of the graph is replaced by an imprecise vertex with minimum height 1 and maximum height 5. Then, each edge is replaced by k vertices at fixed height 3 in the middle of the edge, which are connected to both neighboring vertices. Figure 15 shows an example. Finally, we complete the triangulation by adding dummy vertices at height 5 and triangulating the resulting point set.

Now, a vertex becomes a local minimum if it is assigned a height of at most 3. On the other hand, a fixed vertex (an edge of the original graph) becomes a local minimum is both neighbors are assigned a height of at least 3. If k is large enough, this implies that we can only make one of the two vertices incident to an edge higher than 3. The number of local minima in the final terrain is equal to the number of such vertices with a height higher than 3. These vertices form an independent set in the original graph.

Theorem 6: Minimizing the number of local minima in an imprecise terrain is NP-hard if a local minimum is defined as a vertex without any lower neighbors.

Note, however, that this proof also relies on degeneracy in the input. If we assume that all input heights are different, then the edges incident to a given vertex have different heights, meaning we can put the vertex higher than some of them but lower than others. In this situation, the algorithm from Section 2 (with some small adaptations) can still be used to remove all local minima.

5.2 Minimizing-extrema is still hard when all heights are different

The reduction from P2-MaxCon to our problem depends heavily on making vertices the same height: all red vertices are supposed to be at the same height, and it should be possible to make a connecting path at exactly that height, otherwise there will be more maxima than components. This means that the reduction does not carry through when we require the input to be in general position.

However, we can adapt the construction in Section 4.3 to use different heights at all vertices. The reason is that in the final solution of that problem, all paths of red vertices are routed down (in the y-direction, not in the z-direction) towards a single high vertex at the bottom of the construction, while all paths of blue vertices are routed up towards a single low vertex at the top of the construction. This means we can alter the construction, replacing all points (x, y, z) by a point $(x, y, z - \varepsilon y)$. If ε is small enough, this will make all points with different y-coordinates have different z-coordinates too, while not changing any property of the construction (any pair of neighboring points at the same height now will have the bottom one higher than the top one). Finally, we can make the points with different x-coordinates have different heights as well by simply adding some random noise (even smaller than ε).

Theorem 7: The minimizing-extrema problem in an imprecise terrain with n vertices, for terrains in general position, cannot be approximated in polynomial time within a factor of $O(\log \log n)$, unless $\mathrm{P} = \mathrm{NP}$.

6 Discussion

We have shown that the problem of removing local minima from an imprecise terrain can be solved in $O(n \log n)$ time, while removing local extrema is hard to approximate, even within a factor of $O(\log \log n)$. We have also shown that P2-Con is NP-hard, which constitutes the first result about 2-Disjoint Connected Subgraphs for planar graphs, to the best of our knowledge.

The main remaining open question is whether any constructive results for approximating the problem of minimizing the number of local extrema in imprecise terrains can be approximated, either in the general case or in some special case. Equation 1 suggests that in real terrains, it may be much easier to remove local extrema than what our theoretical results suggest.

Acknowledgments

We thank Jeff Phillips for proposing the problem in Section 4. C.G. is funded by the German Ministry for Education and Research (BMBF) under grant number 03NAPI4 “ADVEST”. M.L. is funded by the U.S. Office of Naval Research under grant N00014-08-1-1015. R.I.S. is supported by the Netherlands Organisation for Scientific Research (NWO).

References

[1] N. Alon, D. Moshkovitz, and S. Safra. Algorithmic construction of sets for k-restrictions. *ACM Transactions on Algorithms*, pages 153–177, 2006.

[2] T. de Kok, M. van Kreveld, and M. Löffler. Generating realistic terrains with higher-order Delaunay triangulations. *Comput. Geom. Theory Appl.*, 36:52–65, 2007.

[3] P. F. Fisher and N. J. Tate. Causes and consequences of error in digital elevation models. *Progress in Physical Geography*, 30(4):467–489, 2006.

[4] C. Gray and W. Evans. Optimistic shortest paths on uncertain terrains. In *Proc. 16th Canadian Conference on Comput. Geom.*, pages 68–71, 2004.

[5] C. Gray, M. Löffler, and R. I. Silveira. Smoothing imprecise 1.5D terrains. In *Proc. Sixth International Workshop on Approximation and Online Algorithms*, pages 214–226, 2009.

[6] J. Gudmundsson, M. Hammar, and M. van Kreveld. Higher order Delaunay triangulations. *Comput. Geom. Theory Appl.*, 23:85–98, 2002.

[7] F. Kammer and T. Tholey. The complexity of minimum convex coloring. In *Proc. 19th International Symposium on Algorithms and Computation*, pages 16–27, 2008.

[8] Y. Kholondyrev and W. Evans. Optimistic and pessimistic shortest paths on uncertain terrains. In *Proc. 19th Canadian Conference on Comput. Geom.*, pages 197–200, 2007.

[9] V. S. A. Kumar, S. Arya, and H. Ramesh. Hardness of set cover with intersection 1. In *ICALP*, pages 624–635, 2000.

[10] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[11] D. Mark. Network models in geomorphology. In M. G. Anderson, editor, *Modelling Geomorphological Systems*, chapter 4, pages 73–97. John Wiley & Sons, 1988.

[12] L. W. Martz and J. Garbrecht. An outlet breaching algorithm for the treatment of closed depressions in a raster DEM. *Computers & Geosciences*, 25:835–844, 1999.

[13] D. Paulusma and J. van Rooij. On partitioning a graph into two connected subgraphs. In *Proc. 20th International Symposium on Algorithms and Computation*, pages 1215–1224, 2009.

[14] R. I. Silveira and R. van Oostrum. Flooding countries and destroying dams. In *Proc. 10th Workshop on Algorithms and Data Structures*, LNCS 4619, pages 227–238, 2007.

[15] A. Temme, J. Schoorl, and A. Veldkamp. Algorithm for dealing with depressions in dynamic landscape evolution models. *Computers & Geosciences*, 32:452–461, 2006.

[16] P. van 't Hof, D. Paulusma, and G. Woeginger. Partitioning graphs in connected parts. *Theoretical Computer Science*, 410:4834–4843, 2009.

[17] D. B. West. *Introduction to Graph Theory*. Prentice Hall, 2nd edition, September 2000.

[18] Q. Zhu, Y. Tian, and J. Zhao. An efficient depression processing algorithm for hydrologic analysis. *Computers & Geosciences*, 32:615–623, 2006.